Homework #3

[ECE30021/ITP30002] Operating Systems

Mission

- Solve problem 1, 2, and 3
  - Solve the problems on Ubuntu Linux (VirtualBox) using vim and gcc.
- Submission
  - Submit a .tgz file containing hw1_3_updated.c, hw3_2a.c, hw3_2b.c, and hw3_3.c on LMS.

    "tar cvfz hw3_*<student_id>*.tgz hw2_3_updated.c hw3_2a.c hw3_2b.c hw3_3.c"

    (Do not copy&past but type the above command)
  - After compression, please check the .tgz file by decompressing in an empty directory.

    "tar xvfz hw3_*<student_id>*.tgz"
- Due date: PM 11:00:00, Apr. 4th

Honor Code Guidelines

- “과제”
  - 과제는 교과과정의 내용을 소화하여 실질적인 활용 능력을 갖추기 위한 교육활동이다. 학생은 모든 과제를 정직하고 성실하게 수행함으로써 과제에 의도된 지식과 기술을 얻기 위해 최선을 다해야 한다.
  - 제출된 과제물은 성적 평가에 반영되므로 공식적으로 허용되지 않은 자료나 도움을 획득, 활용, 요구, 제공하는 것을 포함하여 평가의 공정성에 영향을 미치는 모든 형태의 부정행위는 단호히 거부해야 한다.
  - 수업 내용, 공지된 지식 및 정보, 또는 과제의 요구를 이해하기 위하여 동료의 도움을 받는 것은 부정행위에 포함되지 않는다. 그러나, 과제를 해결하기 위한 모든 과정은 반드시 스스로의 힘으로 수행해야 한다.
  - 담당교수가 명시적으로 허락한 경우를 제외하고 다른 사람이 작성하였거나 인터넷 등에서 획득한 과제물, 또는 프로그램 코드의 일부, 또는 전체를 이용하는 것은 부정행위에 해당한다.
  - 자신의 과제물을 타인에게 보여주거나 빌려주는 것은 공정한 평가를 방해하고, 해당 학생의 학업 성취를 저해하는 부정행위에 해당한다.
  - 팀 과제가 아닌 경우 두 명 이상이 함께 과제를 수행하여 이를 개별적으로 제출하는 것은 부정행위에 해당한다.
  - 스스로 많은 노력을 한 후에도 버그나 문제점을 파악하지 못하여 동료의 도움을 받는 경우도 단순한 문법적 오류에 그쳐야 한다. 과제가 요구하는 design, logic, algorithm의 작성에 있어서 담당교수, TA, tutor 이외에 다른 사람의 도움을 받는 것은 부정행위에 해당한다.
  - 서로 다른 학생이 제출한 제출물간 유사도가 통상적으로 발생할 수 있는 정도를 크게 넘어서는 경우, 또는 자신이 제출한 과제물에 대하여 구체적인 설명을 하지 못하는 경우에는 부정행위로 의심받거나 판정될 수 있다.

Problem 1

- Mission: peer review of HW#2 as a group
  - Sign up study group
    - Section 01: https://docs.google.com/spreadsheets/d/13c8OSrn7-l100vbSHuemXOCIPgH5tM9L3tftIvLdJ34/edit?usp=sharing
    - Section 02: https://docs.google.com/spreadsheets/d/1ozD9CYSYENmA7nza2jXrN4himsghODde25nUyffQDOE/edit?usp=sharing
  - Have an online meeting to review HW#2
    - Review the solutions of other members and share comments to each other.
    - Each member update solution of HW#2 problem 3 (hw2_3_updated.c) individually applying the comments.
      - Mark the modified parts by comments as the next page.
      - If you believe your previous solution was perfect, submit it again and put a comment “My previous solution was perfect, so I didn't modify it!” in the first line.

Problem 1

- Example of updated solution.

```
int main()
{
        int i = 0;

        // previous code
        /*
        <old code>

        */

        // updated code
        <new code>


        return 0;
}
```

Problem 2

- Search the Internet for the following structures and functions.

현재 실행되는 process에 대한 복사본을 만들어 자식 process 생성 / fork() 호출하는 process는 부모, 생성된 process는 자식 → unistd.h (pid_t fork(void))

  - fork()

성공 : 부모 process에서는 자식 process의 PID 값 return / 자식 process에는 '0' return

실패 : return -1

  - wait()

성공 : return 종료된 자식 process의 pid : 자식 process가 종료 될 때까지 기다릴 수 O → sys/wait.h (pid_t wait (int *statloc)

실패 : return -1

  - execl(), execvp()

성공. 이미 다른 program return value x : exec 계열 함수는 다른 program을 실행하고 자신은 종료

실패 : -1

unistd.h

int execvp(const char * file, char *const argv[])

→ P가 있는 경우 환경변수 PATH를 참조하기 때문에 절대경로 입력X

int execl(const char *path, char *const argv[] ···)

→ 절대 경로 입력

  - getpid(), getppid()

pid_t getpid (void) : 함수를 호출한 process 의 ID return

pid_t getppid (void) : 부모 process의 ID return

  - struct timeval

```
struct timeval {
    time_t tv_sec;          // Seconds
    suseconds_t tv_usec;    // Microseconds
}
```

: sys/time.h

  - gettimeofday()

소스코드의 수행 시간 차이를 계산하기 위해 마이크로 단위의 시간 지원 (sys/time.h)

int gettimeofday(struct timeval *tv, struct timezone *tz)

↳ 잘 사용X

  - sleep()

sleep(int second)

: second 후 다음 작업을 수행

Problem 2

- Write a program that creates a child process and measure the running time of the child.
  - Composed of hw3_2a (parent) and hw3_2b

```
$ gcc hw3_2a.c –o hw3_2a          # be careful not to make a typo
$ gcc hw3_2b.c –o hw3_2b          # be careful not to make a typo
$ ./hw3_2a
[parent] before fork(), pid = 5507, ppid = 14483
[child] pid = 5508, ppid = 5507, child_pid = 0
[new child] pid = 5508, ppid = 5507
[new child] 1
[new child] 2
[new child] 3
[parent] pid = 5507, ppid = 14483, child_pid = 5508, running time = 3.002041
```

아무숫자

0 반환시) exec 호출전후

초기다림 sleep(1) x3 호출

어떤 process가 출력한건지 구분

↳ parent wait

Problem 2

- Algorithm (hw3_2a.c)
  - Display pid and ppid using getpid() and getppid()
    - pid: process ID, ppid: parent's pid
  - Get current time using gettimeofday() (start_time)
  - Create a child by fork()
    - pid_t child_pid = fork();
  - In parent
    - Wait for the child to terminate
    - Get current time using gettimeofday() (end_time)
    - Compute the running time of the child from start_time and end_time
    - Display pid, ppid, child_pid, and running time of child
  - In child
    - Display pid, ppid, and child_pid
    - Replace itself by 'hw3_2b' using execl()

Problem 2

- Algorithm (hw3_2b.c)
  - Display pid and ppid using getpid() and getppid()
  - Repeat for i from 1 to 3
    - Display i
    - Sleep for 1 sec. using sleep()

- Message prefixes
  - All messages from parent should be prefixed by “[parent]”
  - All messages from child before execl() (hw3_2a.c) should be prefixed by “[child]”
  - All messages from child after execl() (hw3_2b.c) should be prefixed by “[new child]”

Problem 3

- Search the Internet for the following structures and functions.

cwd; current working directory

include <dirent.h>

- getcwd() → 현재 작업 중인 directory 경로를 얻음. char *getcwd(char *buffer, int maxlen) return < 성공 : 0 / 실패: -1

  file path (buffer) · file에 설정할 접근 mode 값 (maxlen)

- chdir() → 현재 작업 중인 directory 변경 int chdir(const char *dirname) < 성공: 0 / 실패: -1

  변경할 directory name (dirname)

- Write a shell using fork(), wait(), execvp(), and ParseCommand() hw(

```
$ ./hw3_3                                        # bash
Welcome to my_shell!                             # my_shell started
my_shell /home/callee/Courses/ECE321/2022/HW#3> ls
a.out  hello.c  hw3_2  hw3_2a  hw3_2a.c  hw3_2b  hw3_2b.c
hw3_3  hw3_3.c
my_shell /home/callee/Courses/ECE321/2022/HW#3> exit
Bye!
$                                                # returned to bash
```

Problem 3

- Example

```
$ ./hw3_3
                                  getcwd() 사용
Welcome to my_shell!              # my_shell started
my_shell /home/callee/HW#3> ls
hello.c hw3_2a  hw3_2a.c  hw3_2b  hw3_2b.c  hw3_3  hw3_3.c
my_shell /home/callee/HW#3> ls -al
total 92
drwxrwxr-x 2 callee callee  4096  3월 27 15:24 .
drwxrwxr-x 5 callee callee  4096  3월 27 11:37 ..
-rw-rw-r-- 1 callee callee    76  3월 27 11:40 hello.c
…
-rwxrwxr-x 1 callee callee 13008  3월 27 14:31 hw3_3
-rw-rw-r-- 1 callee callee  1802  3월 27 14:32 hw3_3.c
my_shell /home/callee/HW#3> cd ..  use chdir()
my_shell /home/callee> cd HW#3
my_shell /home/callee/HW#3> gcc hello.c
my_shell /home/callee/HW#3> ./a.out
Hello, World!
my_shell /home/callee/HW#3> exit
Bye! 반드시출력
$                                 # returned to bash
```

Problem 3

- **Algorithm (read carefully)**
  - Display a welcome message
  - Repeat
    - □ Display a prompt including current directory
    - □ Read a command line from the user
      - □ fgets(command, MAX_COMMAND, stdin); // define MAX_COMMAND by 512
      - □ command[strlen(command) – 1] = 0; // to trim ‘₩n’
    - □ Parse the command line into arguments using ParseCommand()
    - □ If the command is empty, ignore it.
    - □ If the command is “exit”, break the loop.
    - □ If the command is “cd”
      - □ Change directory to the 1st argument by chdir()
    - □ Otherwise, run the command using a child process
      - □ Create a child process
      - □ In parent ↗ & 있으면 wait X (띄어쓰기)
        - □ If the last argument is not “&”, wait for the child to terminate.
      - □ In child ↗ child가 호출, 명령어 잘못치면 X return value 확인하기
        - □ Run the command by execvp() ↳ child 명시적 X, exit func 호출
        - □ On failure, print “Cannot execute command.” and terminate the child process
  - Display a good-bye message